Syntactic Analysis.
Write a parser for MiniLan (III) - Second part

Theory of Automata and Discrete Mathematics
School of Computer Science
University of Oviedo

1 Introducing the boolean expressions

`printSentence` supports both the arithmetic expressions *arithExpr* and the boolean expressions *boolExpr*: the former has a numerical Double value, the latter has a Java Boolean value. If we take a look to the language's syntax specification, we can see that boolExpr's are just comparisons of two arithExpr's. Let's try to add the boolExpr to our parser.

So, the tasks needed to be solved are:

1. **Introduce the new type of non terminal node:** call it `boolExpr`, and give to it the correct data type.

2. **Introduce the rules:** you must write the rules that produce a boolExpr, including the three cases for ==, > and <. Additionally, write the corresponding Java code for evaluating them and for printing the following message, where x1 and x2 are the values stored in each of the operands and OP stands for the corresponding comparison operator:

   ```
   PARSER:: boolExpr <== x1 OP x2
   ```

   Recall that to compare Double objects you need the *compareTo* method. This method compares the current object with the Double that is given as a parameter. It returns 0 if both are equal, a value greater than 0 in case of current object being greater than than the given as parameter one; otherwise, it returns a negative number.

   ```
   Double a=new Double(8);
   if (a.compareTo(new Double(10))==0) { ...}
   ```

3. **Extend printSentence:** allowing not only to print arithExpr but boolExpr as well. In case of a boolExpr, the message to be printed must be as follows, with XXX being `true` or `false`:

   ```
   PARSER:: printSentence <== PRINT (boolExpr<XXX>)
   ```